

**Japanese Patent Office
Patent Laying-Open Gazette**

Patent Laying-Open No. 9-26876
Date of Laying-Open: January 28, 1997
International Class(es): G 06 F 9/30

(12 pages in all)

Title of the Invention: Method and Apparatus for Multiplatform
Stateless Instruction Set Architecture

Patent Appln. No. 7-305194

Filing Date: October 31, 1995

Priority Claimed: Country: U.S.A.
Filing Date: October 31, 1994
Serial No. 08/332005

Inventor(s): Paul Borrill

Applicant(s): SUN MICROSYSTEMS, INC.

(transliterated, therefore the
spelling might be incorrect)

THIS PAGE BLANK (USPTO)

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平9-26876

(43)公開日 平成9年(1997)1月28日

(51)Int.Cl.⁶
G 0 6 F 9/30

識別記号
3 1 0

庁内整理番号

F I
G 0 6 F 9/30

3 1 0 E

技術表示箇所

審査請求 未請求 請求項の数11 F D (全 12 頁)

(21)出願番号 特願平7-305194
(22)出願日 平成7年(1995)10月31日
(31)優先権主張番号 08/332005
(32)優先日 1994年10月31日
(33)優先権主張国 米国 (US)

(71)出願人 591064003
サン・マイクロシステムズ・インコーポレ
ーテッド
SUN MICROSYSTEMS, IN
CORPORATED
アメリカ合衆国 94043 カリフォルニア
州・マウンテンビュー・ガルシア アヴェ
ニュー・2550
(72)発明者 ボール・ボリル
アメリカ合衆国 95014 カリフォルニア
州・カップチーノ・マドリッド ロード・
10640
(74)代理人 弁理士 山川 政樹

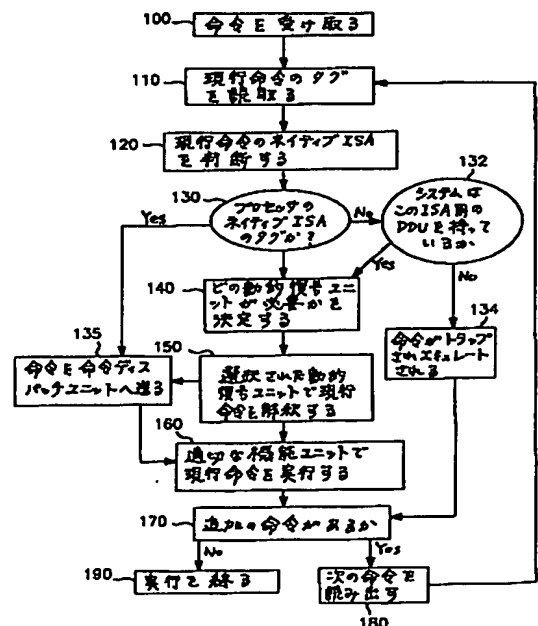
(54)【発明の名称】 マルチプラットフォーム・ステートレス命令セット・アーキテクチャのための方法および装置

(57)【要約】

【課題】 複数の、通常は互換性のない命令セット・アーキテクチャ用に作成された命令を、単一の新しい統一アーキテクチャで実行することができるようにする。

【解決手段】 プロセッサは実行のための命令を受け取ると、命令を検査して、ISAタグから、命令がどの元のネイティブISAに対応しているかを判断する。対応するISAがプロセッサのネイティブVLIW ISAである場合、命令はプロセッサの命令ディスパッチユニットに送られ、次に実行のために少なくとも1つの機能ユニットに送られる。対応するISAがネイティブVLIW ISAではない場合、命令は複数の動的復号ユニット(DDU)の1つに送られる。各DDUは命令を非ネイティブISAからネイティブVLIW ISAに変換する変換ルーチンによって制御される。

プラットフォーム独立コードの実行



【特許請求の範囲】

【請求項1】 プロセッサとそのプロセッサに接続されたメモリとを備え、そのメモリには第1および第2の命令を操作するプログラム命令を含む複数の制御モジュールを記憶し、前記第1および第2の命令はそれぞれ第1および第2の所定の命令セット・アーキテクチャ（ISA）に従って構成されており、第1のISAが前記プロセッサのネイティブISAで、第2のISAが前記プロセッサの非ネイティブISAであるコンピュータ・システム内で前記第1および前記第2の命令を実行するシステムにおいて、

前記プロセッサに結合され、前記第1および第2の命令を受け取る命令ルータと、

命令ルータに結合され、少なくとも前記第1の命令を受け取る命令ディスパッチユニットと、

前記命令ルータに結合され、前記第2の命令を受け取る少なくとも1つの動的復号ユニットと、

前記第1および第2の命令の各命令に結合され、それらの命令に対応するISAを識別するISAタグと、

前記第1および第2の命令セットの前記各ISAタグを読み取るタグ読取り制御モジュールと、

前記各ISAタグに対応するISAを判断するタグ識別制御モジュールと、

プロセッサのネイティブISAに対応するISAタグを有する少なくともいくつかの前記命令を前記命令ディスパッチユニットに送る第1の経路指定制御モジュールと、

所定の非ネイティブISAに対応するISAタグを有する少なくともいくつかの前記命令を前記動的復号ユニットに送る第2の経路指定制御モジュールと、

前記動的復号ユニットで受け取った命令を前記プロセッサの前記ネイティブISAに対応する修正済み命令に変換する変換制御モジュールとを含むシステム。

【請求項2】 前記修正済み命令を前記命令ディスパッチユニットに渡す第3の経路指定制御モジュールをさらに含む請求項1に記載のシステム。

【請求項3】 前記命令ディスパッチユニットに結合され、その命令ディスパッチユニットからの命令を受け取り、その命令を実行する少なくとも1つの機能ユニットと、

前記命令ディスパッチユニットで受け取った命令を前記機能ユニットに渡す第4の経路指定制御モジュールとをさらに含む請求項1に記載のシステム。

【請求項4】 前記命令ルータに結合された命令ローダと、

少なくともいくつかの前記第2の命令を前記ISAタグを含む形式に変換するローダ制御モジュールとをさらに含む請求項1に記載のシステム。

【請求項5】 プロセッサと、そのプロセッサに結合されたメモリと、前記プロセッサに結合された少なくとも

1つの機能ユニットと、前記プロセッサに結合された少なくとも1つの動的復号ユニットと、前記プロセッサにとって非ネイティブである第1の所定の命令セット・アーキテクチャ（ISA）に対応する前記命令を操作する、前記メモリに記憶された前記制御プログラム・モジュールとを有するコンピュータ・システム上で命令を実行する方法において、

前記プロセッサで少なくとも1つの前記命令を受け取るステップと、

10 前記命令を検査して、前記プロセッサのネイティブISAに対応しているかどうかを判断し、対応している場合には前記命令を実行のために前記機能ユニットに渡すステップと、

検査した前記命令が前記第1の所定の非ネイティブISAに関係する場合には、検査した前記命令を前記ネイティブISAに対応する修正された命令に変換するステップと、

前記修正済み命令を実行のために前記機能ユニットに渡すステップとを含む方法。

20 【請求項6】 前記命令受取りステップの前に、少なくとも1つの前記命令を前記ネイティブISAに対応する形式に変換して前記命令を前記非ネイティブISAに対応するものとして識別するステップをさらに含む請求項5に記載の方法。

【請求項7】 命令を検査するステップの後に、前記検査済み命令が、前記プロセッサにとって非ネイティブである第2の所定のISAに対応する場合には、前記検査済み命令をエミュレートする追加のステップを含む請求項6に記載の方法。

30 【請求項8】 プロセッサと、そのプロセッサに結合され、命令を操作するプログラム命令を含む複数の制御モジュールを記憶するメモリとを備え、前記命令は第1および第2の所定の命令セット・アーキテクチャ（ISA）のうちの少なくとも1つに従って構成されており、第1のISAが前記プロセッサのネイティブISAであり、第2のISAが当該プロセッサの所定の非ネイティブISAであるコンピュータ・システムで命令を実行するシステムにおいて、

プロセッサに結合され、前記命令を受け取る命令ルータと、

40 前記命令ルータに結合され、前記命令のうちの少なくとも第1のサブセットを受け取る命令ディスパッチユニットと、

前記命令ルータに結合され、前記命令のうちの少なくとも第2のサブセットを受け取る少なくとも1つの動的復号ユニットと、

複数の前記命令のそれぞれに対応し、複数の命令のそれぞれのネイティブISAを識別するISA識別子と、

前記複数の命令のそれぞれが当該命令のネイティブISAであると判断するISA識別制御モジュールと、

命令の前記第1のサブセットの少なくともいくつかを前記命令ディスパッチユニットに送り、命令の前記第2のサブセットの少なくともいくつかを前記動的復号ユニットに送る経路指定制御モジュールと、

前記動的復号ユニットで受け取った命令を前記プロセッサの前記ネイティブISAに対応する修正された命令に変換する動的復号ユニット制御モジュールとを含むシステム。

【請求項9】 前記命令ディスパッチユニットに結合された複数の機能ユニットと、

前記命令ディスパッチユニットで受け取った命令を実行のために少なくとも1つの前記機能ユニットに経路指定する命令ディスパッチユニット制御モジュールとをさらに含む、請求項8に記載のシステム。

【請求項10】 前記プロセッサに結合され、前記プロセッサへの入力と前記プロセッサによる実行のために前記命令を受け取る命令ローダと、

複数の前記命令のそれぞれを前記プロセッサの前記ネイティブISAに対応する形式に変換する命令変換制御モジュールとをさらに含む請求項8に記載のシステム。

【請求項11】 前記プロセッサにとって非ネイティブである第2の所定のISAをエミュレートする命令エミュレーション制御モジュールをさらに含む、請求項8に記載のシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、コンピュータ処理装置で実施され、複数の異なる命令セット・アーキテクチャ（ISA）に準拠する命令を実行するシステムに関し、具体的には様々なISAの前記命令をステートレスで実行する処理装置に係わる。

【0002】

【従来の技術】2進変換またはエミュレーションによって様々なタイプの命令に対応することができる複数ISAプラットフォームがいくつか提案されている。2進変換を使用すると、特定の命令セット・アーキテクチャB（たとえばx86プロセッサ用）に準拠する命令を、まずISA Aが理解する命令に2進変換することによって、第2の命令セット・アーキテクチャA（たとえばSPARC/IEEE標準1754プロセッサ用ISAに準拠したもの）で実行することができる。次にそれらの命令を、もともとISA A用に作成された命令であるかのように実行する。

【0003】この手法は、ISA B用のプログラムの一部をISA-Aプロセッサで実行する場合であっても、その前にまずプログラム全体を2進変換しなければならないという点で、かなりのオーバーヘッドが伴う。これによって、ユーザが命令を処理できるまでに遅延が生ずるだけでなく、命令の処理を拘束し、2進変換が完了するまでターゲット・プロセッサが使用不能になる。

さらに、所与の時間に変換中のコードの小部分のみを必要とする場合、その部分にアクセスするまでにコード全体を変換しなければならないという点で非常に非効率的である。

【0004】さらに、2進変換は元の命令をコピーすることを必要とし、ユーザのライセンスまたはユーザが変換を行う地域の法律によってコピーが許されている場合と禁止されている場合がある。さらに、変換された命令を記憶しなければならないため、ユーザはコードの2つのコピーを保存することを余儀なくされる。変換されたコピーを廃棄する場合は、そのコードを後で使用するために、ISA-Aプロセッサ上で実行するように再度変換しなければならない。

【0005】ISA Aを使用するシステムでISA B用に作成されたコードを実行する他の手法としてはエミュレーションがある。これは、ソフトウェア・エミュレーションとハードウェア・エミュレーションのいずれかである。ソフトウェア・エミュレーションでは、通常であればISA A用プロセッサで認識しない命令を入力し、それを同等のISA-A命令に変換し、同等の命令をAプロセッサで実行する必要がある。これは、AプロセッサのモードをISA Bをエミュレートするように設定し、ISA-B命令を実行し、次にAプロセッサを再びAモード、すなわちネイティブISAにリセットすることによって行う。

【0006】モード変更は、プロセッサ（またはソフトウェア・エミュレーション・プログラム）に対して、次に続くデータ・ブロック（モードのリセットまで）をISA-B命令として解釈するように命令する特別なプログラムによって行われる。たとえば、SPARCプロセッサに対して、次に続くデータ・ブロックが実際にはx86命令を含むことを伝える。SPARCプロセッサは、各データ・ワードを1つずつ命令として解釈する。そのデータ・ブロックを命令として実行し終わると、モード設定プログラムがプロセッサのモードをそのネイティブ（たとえばSPARC）モードにリセットする。

【0007】このようにしてISA-B命令を正常に実行することができるが、これは、特にモード変更のオーバーヘッドが重大な量となるコードの短いセクションの場合には、非効率的で時間のかかる手法であり、ISA-A命令とISA-B命令とをコードの1つのブロック内で混在させることはできない。すなわち、異なるISA A用のコードが現れるたびに、Aプロセッサのモードをリセットしなければならず、必然的にかかなりのサイクルタイム・オーバーヘッドが生ずる。プロセッサAでのISA-Bコードの実行中に、ネイティブISA-A命令の実行を必要とする割込みを受け取った場合には特にそうである。その場合、ISA-A命令をトラップしてISA-Bコードの実行を中断しなければならず、システムは正しいモードになっているかどうかを検査し、正し

いモードではないためにモードをモードAにリセットしなければならない。次に、トラップした命令を実行した後、システムはモードBにリセットしてISA-Bコードの実行を再開する。これは法外なサイクル数を消費する。

【0008】ハードウェア・エミュレーションは、1つまたは複数の動的復号ユニット（ネイティブ・プロセッサの内蔵部品とするか、またはアドオン・モジュールとすることができる）を使用して、ソフトウェア・エミュレーションよりも高速で実行することができる。従来のハードウェア・エミュレーションは、ソフトウェア・エミュレーションと同様に、モード設定プログラムを使用して、所与の命令セットがどのタイプのISAをエミュレートするかをプロセッサが知り、命令を適切な動的復号ユニットに送ることができるようになっている。実行速度は速くなるが、モードのリセットと、トラップと、割込みというオーバーヘッド非効率要素はソフトウェア・エミュレーションの場合と同じである。

【0009】複数命令セット・アーキテクチャに対応するその他の手法は、増分コンパイル、すなわち1回にプログラムの一部をコンパイルする方法である。これは、プログラムの1部分だけを必要とする場合にプログラム全体をコンパイルする必要がなく、所与の設定でコンパイル時間をかなり節約することができるという利点がある。この手法は、コードを修正するたびに再コンパイルしなければならないため、自己修正コードが効率的に処理されないという明らかな欠点がある。

【0010】同じ命令が繰り返し実行されるループでは、ソフトウェア・エミュレーションまたはハードウェア・エミュレーションを行うと、エミュレーション・プロセスはコマンドが現れるたびに各コマンドを新たに解釈するため、同じコード・ブロックが何度も再コンパイルされることになる。増分コンパイルは、コードの自己修正が起こったとき、同じコンパイルが繰り返される可能性がある。このコンパイルの反復は相当なプロセッサ・サイクルの無駄である。

【0011】

【発明が解決しようとする課題】SPARC、x86、PowerPCの各ISAおよびそれぞれのオペレーティング・システム（Solaris、DOS、MacOS）など、市場における複数命令セット・アーキテクチャの絶え間のない開発により、ユーザがこれらの様々な命令セット・アーキテクチャ用に開発されたアプリケーションを単一のハードウェア・プラットフォーム上で、不要な命令ブロックのコンパイルや、命令の非効率的な複数回コンパイル、モード設定に伴うオーバーヘッドがない方式で実行できることがますます重要になっている。複数命令セット・アーキテクチャに効率的に対応できるだけでなく、異なる命令セットの命令が入ったプロセスを混在させることができることによって、様々

なISA用に作成された多くのプログラムのうち最良のものを利用することができるシステムがあれば、特に有用であろう。

【0012】

【課題を解決するための手段】本発明は、複数の、通常は互換性のない命令セット・アーキテクチャ用に作成された命令を、単一の新しい統一アーキテクチャで実行することができるようにする。所与の32ビット・アーキテクチャ用の命令を、「外来」コードのネイティブISAを示すISAタグを構成する所定のビット・ブロックが含まれたもう1つの32ビット・ワード、すなわちコードが書き込まれた32ビット命令セット・アーキテクチャと結合する。この64ビット命令ストリーム全体を「ホーム」プロセッサが受け取り、そのISAタグに従って各命令を実行する。

【0013】したがって、ホーム・プロセッサは64ビット命令セット・アーキテクチャを使用し、前述のようにして任意の32ビット・アーキテクチャの命令を、再コンパイルやソフトウェア・エミュレーションなしでリアルタイムで実行することができる。これによって、前述の方法に関するオーバーヘッドが減少するほか、プログラマーは、インポートされる各コード・ブロックに適切なタグを付けるだけで、1つのアプリケーションで複数のISAのサブルーチン、ソフトウェア・モジュール、およびオブジェクトを自由に利用することができる。本発明は、VLW（超長命令語）アーキテクチャに特に適しており、このアーキテクチャによって真のステートレス複数ISAシステムが実現される。

【0014】

【発明の実施の形態】本発明のシステムは、プロセッサ30とメモリ40を有する中央演算処理装置（CPU）20を備えた図1に示すシステム10のような既存のコンピュータ・システムで実施するように設計されている。通常、たとえばディスク・ドライブやテープドライブ、CD-ROMなど、記憶媒体50のような1つまたは複数の記憶媒体を使用する。CPU20には命令ローダ65を介して入出力装置60が接続されており、これにはキーボード、モニタ、マウスなどの入出力周辺装置が含まれる。したがって本発明のシステムは、後述の例外を除き、従来のハードウェアを使用して実施することができる。

【0015】本明細書では、本発明について、1つのプラットフォームで複数の異なる命令セット・アーキテクチャ用に作成された命令をステートレスに実行する実施形態、具体的には、64ビット・プラットフォームで32ビット命令を実行する実施形態で説明する。元の非ネイティブ命令セット・アーキテクチャよりも長い命令を使用して、1つのプラットフォーム上でのどのような命令セットの実行にも等しく適用可能である。これは、元のサイズが異なる様々なセットであっても可能である。

7

本発明は、メモリ40に記憶されているオペレーティング・システム、すなわち、後述する方法の各ステップを実行するためにメモリに記憶された制御命令、一連の命令、あるいは1つまたは複数の制御命令を含むプログラム・モジュールの形態で実施するのが最もよい。本明細書では、「制御モジュール」とは、本発明の1つまたは複数のステップを実行するために構成された、任意のプログラム、命令のセット、プログラム・ルーチンなどを指す(図7を参照)。

【0016】図2に、それぞれが1つまたは複数の制御命令を含んだ制御命令、一連の制御命令、またはプログラム・モジュール70ないし78の、これらは32ビットの命令である、プロセッサ80への流れを図示する。前述のように、プロセッサ80のネイティブISA用に作成された命令でない場合、これらの命令を実行するにはいくつかの異なる方法がある。命令70ないし78の上下の点は、所与のプログラム、ルーチン、または同様のものの中には一般にさらに多くの命令があることを示している。

【0017】図3に、本発明のプロセッサ90に送られる、命令70ないし78が変形された命令70Aないし78Aの流れを図示する。各命令には、図3ではビット32ないし63として図示されている追加の32ビットが付加されている。実施形態に示すように、上位3ビット(またはその他の所望のビット数)は、各命令のネイティブISAを示すタグを構成する。3ビットによって、8つの異なる命令セット・アーキテクチャが識別されるため、タグT0ないしT8(それぞれ命令70Aないし78Aに対応する)のビット数は、対応する命令セット・アーキテクチャの数によって異なる。

【0018】図4(図5も参照)のボックス100で、プロセッサ90が命令を受け取ると、命令は命令ルータ205に受け入れられ、それがネイティブISA(この例ではVLIWなど)の命令である場合、命令ルータは命令ディスパッチユニット200に直接送られる。受け取った命令が、DDUを使用することができる所定のいくつかのISAのうちの1つに関係する命令である場合は、命令ルータ205によって適切なDDUに送られ、命令ディスパッチユニット200が認識することができる命令に変換される。いずれの場合も、命令は次に、実行のために適切な機能ユニット(280など)に送られる。機能ユニット(ここでは例示のためにそのうちの3つが示されているが、任意の数とすることができる)としては、浮動小数点ユニット、フェッチユニット、分岐ユニット、算術演算ユニットなどがある。

【0019】受け取った命令の適切な経路指定は次のようにして行われる。ルータ205で命令が受け取られると、それに対応するタグが読み取られ、その命令のネイティブISAが判断される(ボックス110および120参照)。タグは事前割り当てされた意味を有する。た

8

たとえば、000はプロセッサ90による変換が不要なネイティブ・プロセッサのISAを指すものとしてすることができる。表1にコード体系例を示す。

表1

コード	ネイティブ・プロセッサ
000	VLIW
001	SPARC
010	PowerPC
011	x86
100	(プロセッサ5)
101	(プロセッサ6)
110	(プロセッサ7)
111	(プロセッサ8)

【0020】プロセッサ90が命令70Aを受け取ると(図3および図5参照)、プロセッサは、その命令が、この例ではネイティブVLIW命令、すなわちシステムのVLIWプロセッサ用とコードされた命令であると判断する。図4のボックス130を参照されたい。命令70Aは、図4のボックス135で示すようにルータ205(図5参照)によって命令ディスパッチユニット200に送られ、実行のために適切な機能ユニット280、290、または300に送られる(ボックス160)。

【0021】次にシステムは、追加の命令があるかどうかを判断し(ボックス170)、ある場合には、次の命令が読み取られ(ボックス180)、ステップ110に戻って次の命令のタグが読み取られる。

【0022】次の命令は、この時点では命令71Aであり、ネイティブVLIWプロセッサ用にコードされているものと判断され、したがって命令70Aと同様にして実行される。これは命令72Aについても行われ、次に命令73Aが現れると、そのタグがステップ110で前述のように読み取られる。

【0023】ステップ120で、命令73AのネイティブISAがサン・マイクロシステムズ社のSPARC ISAであることが判断される。したがって今度はステップ132に進み、識別されたISAのDDUがシステムに組み込まれているかどうか判断される。答えが肯定の場合ステップ140に進む。否定の場合はステップ134に進み、トラップされてソフトウェア・エミュレーション・ステップでエミュレートされる。

【0024】この例で、システムに適切なDDUが存在するものとする、命令はルータ205によってその適切な動的復号ユニット210ないし270に送られる。3ビットのタグによって対応することができる1個のネイティブ・プロセッサと7個の「非ネイティブ」プロセッサの場合、7個の動的復号ユニット(DDU)が存在することが可能になる。この実施形態の動的復号ユニットの数が7個より少ない場合、残りのISAはトラップしてソフトウェアでエミュレートすることができ、したがってハードウェア実行速度とオンチップ資源の経済性

との間で効果的なトレードオフが可能になる。この特徴については以下で詳しく説明する。

【0025】DDUの機能は、受け取った命令を、ネイティブ・プロセッサ90が認識できる命令に変換することである。これには、命令を非ネイティブISAからネイティブISAに1対1で変換し（これは参照テーブルによって行うことができる）、複合命令をいくつかのネイティブISA命令に変換し（たとえば複合x86命令はいくつかのVLIW命令に変換することができる）、非ネイティブISAの条件コードをネイティブISAの条件コードとして実行することなどが必要である。様々なISA用のアプリケーションを開発する当業者なら、このようなDDUを簡単に実施することができよう。

【0026】本発明で実行される機能およびステップは、前述のように、メモリ40（図7参照）に記憶された個別の対話型命令制御モジュールとして実施されるのが好ましい。図7に示すモジュールの機能は以下の説明から明らかになる。数個のモジュールを結合して1つのモジュールとすることができ、本発明の個別に識別可能な任意の機能のために必要であればその他のモジュールも組み込むことができる。

【0027】したがって、図4のボックス140で、命令73AをDDU1などの適切なDDU（図5の210）に送るべきかどうか判断される。図4のボックス150で、DDU1への命令の送信と、DDU1での解釈が示されている。DDU1は、命令を1つまたは複数のVLIW命令に変換し、図4のボックス135に示されているように命令ディスパッチユニット200（図5）に渡す。ディスパッチユニット200は必要に応じて、命令を実行のために適切な機能ユニットに渡す（ボックス160）。

【0028】ボックス170で、命令ストリーム内に追加の命令があることがわかり、ステップ180に進んで命令74Aが読み取られる。ボックス110でそのタグが読み取られ、この場合もSPARC命令であると判断される。すなわち、そのタグ「001」（図3）から、DDU1に送るべきであると判断される。

【0029】次の2つの命令75Aおよび76AはネイティブVLIW命令（コード「000」を有する）であり、したがって命令ルータ205は命令ディスパッチユニット200に渡す。命令77Aは、「110」というタグによってx86プロセッサ（前記の表1参照）に属するものであることがわかり、ルータ205によって適切なDDU、たとえばDDU3（図5の230）に送られる。

【0030】命令78AもネイティブVLIWコマンドであることがわかり、ディスパッチユニット200（図5参照）に直接送られる。それ以降の命令もそれぞれ同様にして扱われる。

【0031】このシステムによって、任意の多数の命令

セット・アーキテクチャ用に作成されたコードの命令を動的に（リアルタイムで）実行する真にステートレスな方式が実現されることがわかる。これによって、プログラマーは1つのアプリケーションで様々なISAのルーチン、モジュール、およびオブジェクトを使用して、きわめて柔軟性のある方式で、所望の場合は行単位で、アプリケーションを作成することができる。そのように作成した場合、実行時に処理上の不利はなく、事前コンパイルまたはモード設定に伴うオーバーヘッドもない。

【0032】図5の命令ルータ205、命令ディスパッチユニット200、DDU210ないし270および機能ユニット280ないし300は、通常はプロセッサ90の内蔵部品となる。しかし、ユーザの選択システム・アーキテクチャによっては、これらのいずれも別個の要素として設計することができる。本明細書では、これらがプロセッサの内蔵部品であるか、特定の場合には主プロセッサとは別個のモジュールであるかを問わず、これらを一般にプロセッサに「結合された」ハードウェア・モジュールと呼ぶ。後述するように、ルータ205およびユニット200ないし300はすべて、メモリに記憶されている命令または命令モジュールによって制御される。これらの命令または命令モジュールは、着信命令を必要に応じて受信、検査、経路指定、変換およびその他の方法で操作する。

【0033】図5に示すDDUは、図のようにプロセッサに組み込むか、または後で付加できるプラグイン・モジュールとすることができる。したがって、本発明のシステムは、まず最初に所与の数のISAに対応するように構成することができ、モジュールを追加してネイティブ・システムのISAをそれに応じて再構成することによって、その後の命令セット・アーキテクチャにも対応することができる。

【0034】上記で示唆したように、本発明のシステムは従来のシステムと組み合わせて最大限の柔軟性を得ることができる。たとえば、使用可能なDDUがない所与のISA用のコードを実行したい場合は、そのコードのために図4のボックス132ないし134で示してあるように、従来のソフトウェア・エミュレーションを使用することができる。これらのボックスでは、システムがDDUを使用することができない命令のために従来のソフトウェア・エミュレーションが実行される。したがって、入力コードにはソフトウェア・エミュレートを行う所与のISAのための命令ブロックを組み込むことができると同時に、残りの命令には前述のようにネイティブISAを示すためのタグを付けることができる。同様に、2進変換またはハードウェア・エミュレーションを、本発明を利用するように構成された命令とともに使用することもできる。ただし、所与のISAのためにDDUを使用することができれば、前述のように本発明の実施形態でそれを使用することは簡単なことである。

め、後者の方法が用いられることは少ないと思われる。

【0035】図6に、所与のISA用の命令ブロックを、本発明のシステムで実行するのに適した図3に図示する構造に従った命令に変換する方法を示す。この方法は、コンピュータ・システムのメモリに記憶された命令によって行われ、図1に示すローダ65を使用して実行される。

【0036】ステップ310（図6参照）で非ネイティブ命令が受け取られてローダ65に入れられる。ローダにはそれらの命令がどのISA用に作成されたかに関する情報が提供される。各命令には、図3に図示する命令ストリームの左側に示されている32ビット構造体（ビット32ないし63）が付加され（ステップ320）、それぞれの場合について非ネイティブISAの適切なタグが入れられる（ステップ330）。命令はこれによって修正され、64ビット命令に変換されて、ステップ340で修正済みの形式で記憶される。これで命令は前述のようにして実行することができる。

【0037】このような非ネイティブ・コードのブロックが変換され、実行されるとき、本発明のシステムによって適切なアドレス指定が自動的に行われる。図2で、命令70はアドレス「n」、命令71はアドレスn+4（8バイト・ワードとした場合）のようになっているとする。変換後、命令は図3に示す構造になり、変換された命令70Aは（VLIW）アドレス「n」を有するが、命令71Aはn+4ではなくアドレスn+8を有し、命令72Aはn+8ではなくアドレスn+16を有するようになる。これは、特に相対アドレスを使用する場合には問題になることがあるため、本発明のシステムは、32ビット命令セットからの命令の変換時に、64ビット命令セットによって占有される追加アドレス空間を隠すユニットを備える。これは、各アドレスが1バイトではなく2バイトを識別することができるようにすることによって行うことができる。これは、ネイティブISAには見えることになるが、外来ISAには、各アドレスによって識別される記憶バイトがそれ以上あることはわからないため、64ビット命令フィールドの上位32ビットは外来ISAから隠される。

【0038】元々の非ネイティブ・コードは、図6に関して述べたようにして修正されると、正常に実行するためのそれ以上の修正は必要としない。所与のアドレス指定体系にとって、本発明のシステムのためにアドレスを適切に変換するのに必要なコードを生成することは簡単なことである。たとえば、この例では、値（n+Z）を持っている前のアドレスが、値（n+2・Z）を持つアドレスに変換される。ここでZは0、4、8、12などのオフセットであり、すなわち、オフセットは0、8、16、24のように2倍になる。

【0039】まずリアルタイムでDDUによって変換することにより、すべての命令がほとんどネイティブ命令

と同様に実行されるため、非ネイティブ・プラットフォーム上で自己修正命令を実行するという前述の問題は、本発明によって解決されることが認められよう。したがって、コード修正は、2進変換またはエミュレーションで生ずる再コンパイルやその他のオーバーヘッドなしで、すべて自動的に行われる。

【0040】これらの64ビット命令が占める追加空間によって、命令キャッシュ内の占有メモリ帯域幅および空間の量が増えることになる。しかし、VLIWプロセッサはいずれにしてもより大きなレジスタ・ファイルへのアクセスおよび命令符号化を可能にするために、32ビットよりもはるかに大きな命令を必要とするため、これが問題になることはないと思われ、しかも命令が示す一時的、空間的局所性により、命令参照の大半がオンチップ・キャッシュにとどまり、メモリへの追加帯域幅が使い果たされることはない。したがって、本発明の命令の大きさの増大によるキャッシュ・ミスの増加は認めうるほどはないと同時に、直接マルチISAシステムによってかなりのサイクルが節約される。

【図面の簡単な説明】

【図1】 本発明を実施することができるコンピュータ・システムのブロック図である。

【図2】 プロセッサへの32ビット命令の流れを示す図である。

【図3】 プロセッサへの本発明の64ビット命令の流れを示す図である。

【図4】 本発明の好ましい方法を示すフロー・チャートである。

【図5】 本発明を実施するための、プロセッサ内の動的復号ユニットおよび機能ユニットを図示したブロック図である。

【図6】 従来の命令のブロックを、本発明に従って実行するための命令に変換する方法を示すフロー・チャートである。

【図7】 本発明の好ましい実施形態の命令制御モジュールを図示したブロック図である。

【符号の説明】

10 システム

20 中央演算処理装置

30 プロセッサ

40 メモリ

50 記憶媒体

60 入出力装置

65 命令ローダ

70 制御命令

70A 命令

80 プロセッサ

90 プロセッサ

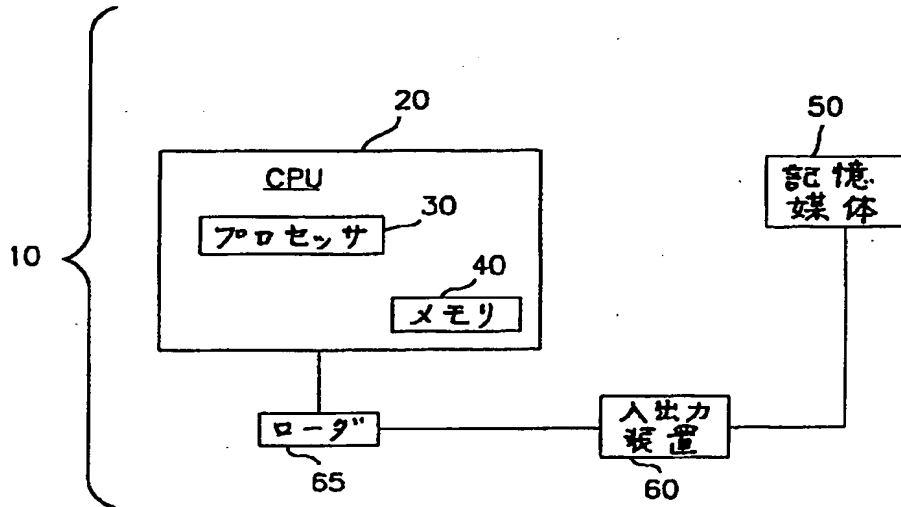
200 命令ディスパッチユニット

210 動的復号装置

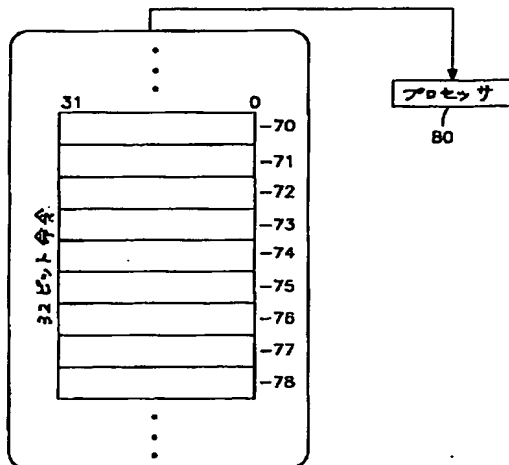
205 命令ルータ

* * 280 機能ユニット

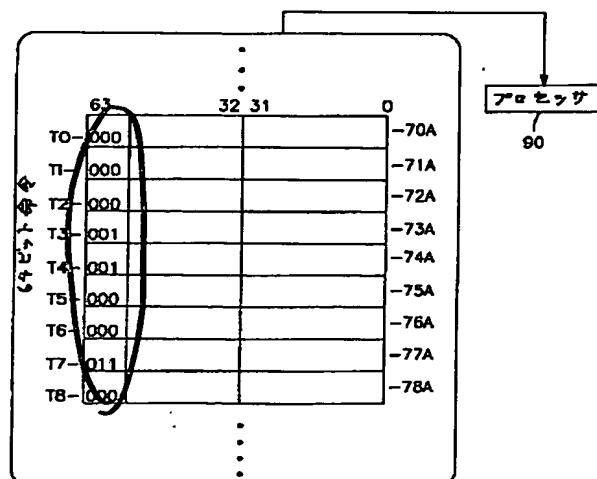
【図1】



【図2】

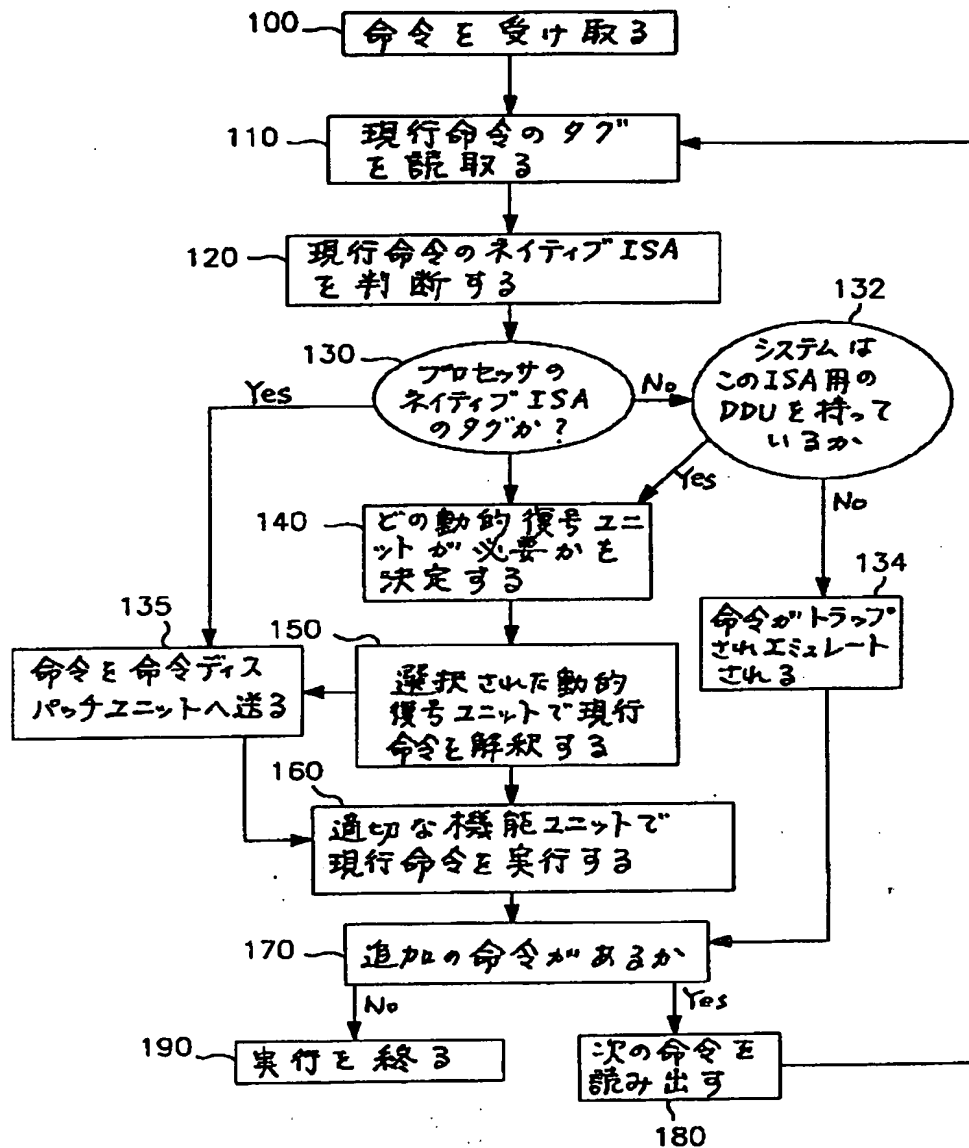


【図3】

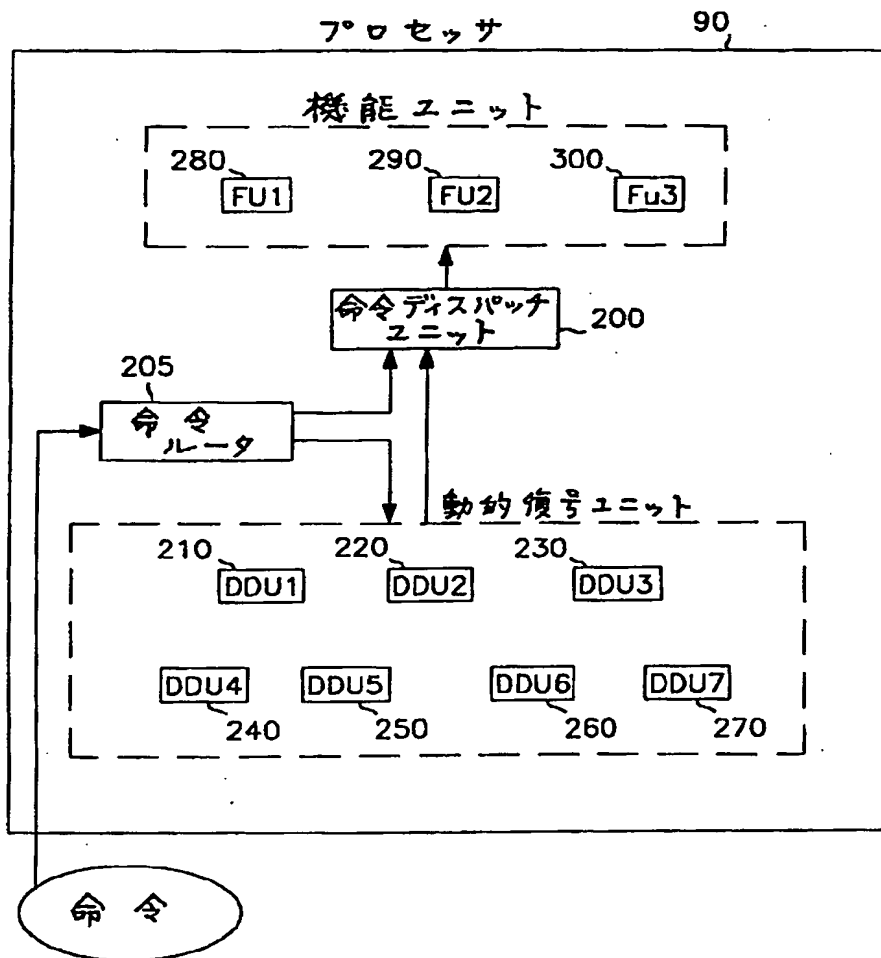


【図4】

プラットフォーム独立コードの実行

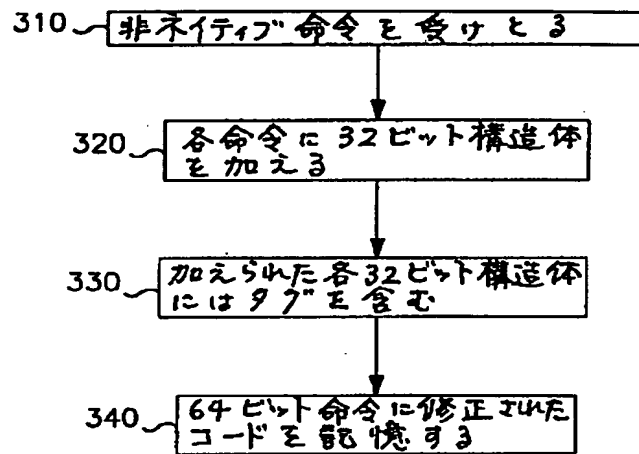


【図5】



【図6】

OS-A コードから プラットフォーム独立コードへの変換



【図7】

